

1                    **DATA PROCESSING SYSTEMS**

2            **FIELD OF INVENTION**

3        The present invention is directed to data processing  
4        systems. More particularly, it is directed to systems,  
5        methods and apparatus for introducing security measures to  
6        data processing systems.

7            **BACKGROUND OF THE INVENTION**

8        A data processing system typically comprises a central  
9        processing unit (CPU), an input/output (I/O) subsystem, and  
10       a memory subsystem, all interconnected by a bus subsystem.  
11       The memory subsystem typically comprises random access  
12       memory (RAM), read only memory (ROM), and one or more data  
13       storage devices such as hard disk drives, optical disk  
14       drives, and the like. The I/O subsystem typically comprises:  
15       a display; a printer; a keyboard; a pointing device such as  
16       a mouse, tracker ball, or the like; and one or more network  
17       connections permitting communications between the data  
18       processing system and one or more similar systems and/or  
19       peripheral devices via a data communications network. The  
20       combination of such systems and devices interconnected by  
21       such a network may itself form a distributed data processing  
22       system. Such distributed systems may be themselves  
23       interconnected by additional data communications networks.  
24       In the memory subsystem is stored data and computer program  
25       code executable by the CPU. The program code includes  
26       operating system software and application software. The  
27       operating system software, when executed by the CPU,  
28       provides a platform on which the application software can be

1   executed. The operating system software has a core or kernel  
2   of code in which the basic functions of the operating system  
3   software are defined.

4   A problem associated with data processing systems is that of  
5   security. In particular, it is becoming increasingly  
6   difficult to determine with any degree of certainty that a  
7   data processing system actually has the properties it is  
8   believed to have. This difficulty arises because data  
9   processing systems, and particularly the operating systems  
10   therein, are becoming increasingly general purpose,  
11   configurable, and reconfigurable in nature. The  
12   administrative state of a data processing systems can be  
13   varied from one moment to the next based on an  
14   administrative action. Specifically, the administrative  
15   state of a data processing system is defined by the  
16   combination of software and data present in the machine. The  
17   software may include binary files, patches, applications,  
18   and the like added to and deleted from the system from time  
19   to time via one or more administrative actions. An  
20   administrative action such as the addition or deletion of  
21   software in the system can thus be regarded as a change in  
22   the state of the system. Many data processing systems can be  
23   placed into a corrupt state by users and/or system  
24   administrators with or without proper authorization. This  
25   form of corruption is difficult to detect. It would be  
26   desirable to make such corruption easier to detect.

27   Many data processing networks employ intrusion detection and  
28   diagnosis (IDD) systems. These IDD systems are typically  
29   data processing systems resident on the network and  
30   dedicated to intrusion detection and diagnosis. It will be  
31   appreciated that detection of corruption is important in the  
32   field of IDD. Most intruders do not want to be detected.  
33   Thus, administration tools employed in IDD systems are among  
34   the first to be attacked.

1 Cracker tools allow hackers, crackers, or other attackers to  
2 selectively hide files, processes, and network connections  
3 in an individual host data processing system. An example of  
4 a conventional cracker tool is known as "rootkit". Rootkit  
5 replaces Unix system commands used for investigation, such  
6 as ls, ps, netstat, and ifconfig, with so-called Trojan  
7 horse versions that hide the activities of an attacker.  
8 Conventionally, such Trojan horses have been identified by  
9 calculating, storing, and comparing databases of  
10 cryptographic checksums of system binaries. However, recent  
11 versions of "rootkit" include Trojan horse versions of the  
12 programs employed to generate and compare the checksums.  
13 Attackers have recently begun to employ loadable kernel  
14 modules to introduce Trojan horses to data processing  
15 systems. A kernel is difficult to inspect when running.  
16 Thus, Trojan horse modules therein remain undetected. A  
17 typical defense against such Trojan horse modules is to  
18 prevent kernel modules from being loaded. However, system  
19 functionality is then limited.

20 There is growing interest in releasing computer software in  
21 source code form under "open source" licenses such as the  
22 "General Public License" and the like. Such releases  
23 facilitate creation of Trojan horses, particularly when the  
24 software in question is operating system software. The  
25 detection of Trojan horses is therefore of increasing  
26 importance. IDD systems are an early target for infiltration  
27 by Trojan horses. Here, the attacker typically alters the  
28 IDD system in such a manner that it appears to continue  
29 functioning normally, but in reality hides the attacker's  
30 activities.

31 Conventional security schemes for data processing systems  
32 include secure logging schemes and forward secrecy of  
33 digital signatures.

1 Secure logging schemes are directed to the protection of  
2 integrity, secrecy, and authenticity of records of data  
3 processing events. The schemes may be employed in  
4 maintaining quality standards of event logging. In general,  
5 secure logging schemes assume the existence of a secure  
6 logging host data processing system and operate in  
7 dependence on a combination of message or stream encryption,  
8 hash chaining, authentication codes, and one way key  
9 evolution.

10 Forward secrecy of digital signatures is directed to  
11 limiting damage to compromised signature keys. In operation,  
12 forward secrecy of digital signatures provides a series of  
13 signing keys:

14  $SK_0 \Rightarrow SK_1 \Rightarrow SK_2 \Rightarrow \dots$

15 So that  $SK_{n+1}$  is a derivation of  $SK_n$ , and that verification  
16 of a signature does not require distribution, traversal, and  
17 verification of a chain of keys.

## 18 SUMMARY OF THE INVENTION

19 In accordance with the present invention, there is now  
20 provided systems apparatus and methods for detecting an  
21 attack on a data processing system. An example method  
22 comprising, in the data processing system: providing an  
23 initial secret; binding the initial secret to data  
24 indicative of an initial state of the system via a  
25 cryptographic function; recording state changing  
26 administrative actions performed on the system in a log;  
27 prior to performing each state changing administrative  
28 action, generating a new secret by performing the

1 cryptographic function on a combination of data indicative  
2 of the administrative action and the previous secret, and  
3 erasing the previous secret; evolving the initial secret  
4 based on the log to produce an evolved secret; comparing the  
5 evolved secret with the new secret; determining that the  
6 system is uncorrupted if the comparison indicates a match  
7 between the evolved secret and the new secret; and,  
8 determining that the system is corrupted if the comparison  
9 indicate a mismatch between the evolved secret and the new  
10 secret.

11 From another aspect, there is now also provided a data  
12 processing system comprising: a processor; a memory  
13 connected to the processor; and, detection logic connected  
14 to the processor and the memory, the detection logic, in  
15 use: providing an initial secret; binding the initial secret  
16 to data indicative of an initial state of the system via a  
17 cryptographic function; recording state changing  
18 administrative actions performed on the system in a log;  
19 prior to performing each state changing administrative  
20 action, generating a new secret by performing the  
21 cryptographic function on a combination of data indicative  
22 of the administrative action and the previous secret, and  
23 erasing the previous secret; evolving the initial secret  
24 based on the log to produce an evolved secret; comparing the  
25 evolved secret with the new secret; determining that the  
26 system is uncorrupted if the comparison indicates a match  
27 between the evolved secret and the new secret; and,  
28 determining that the system is corrupted if the comparison  
29 indicate a mismatch between the evolved secret and the new  
30 secret.

31 The present invention advantageously also provides methods  
32 for cryptographic entangling of state and administration in  
33 a data processing system that permits detection of Trojan  
34 horses within the system.

1     **BRIEF DESCRIPTION OF THE DRAWINGS**

2     The invention and its embodiments will be more fully  
3     appreciated by reference to the following detailed  
4     description of advantageous and illustrative embodiments in  
5     accordance with the present invention when taken in  
6     conjunction with the accompanying drawings, in which:

7     Fig. 1 shows a block diagram of an example of a data  
8     processing system embodying the present invention;

9     Fig. 2 is a flow diagram associated with the data processing  
10    system;

11    Fig. 3 is another flow diagram associated with the data  
12    processing system;

13    Fig. 4 is yet another flow diagram associated with the data  
14    processing system;

15    Fig. 5 is a further flow diagram associated with the data  
16    processing system; and,

17    Fig. 6 is a block diagram of a data processing system in a  
18    layer format.

19    **DETAILED DESCRIPTION OF THE INVENTION**

20    The present invention provides methods, systems and  
21    apparatus for detecting an attack on a data processing  
22    system. An example method comprising, in the data

1 processing system: providing an initial secret; binding the  
2 initial secret to data indicative of an initial state of the  
3 system via a cryptographic function; recording state  
4 changing administrative actions performed on the system in a  
5 log; prior to performing each state changing administrative  
6 action, generating a new secret by performing the  
7 cryptographic function on a combination of data indicative  
8 of the administrative action and the previous secret, and  
9 erasing the previous secret; evolving the initial secret  
10 based on the log to produce an evolved secret; comparing the  
11 evolved secret with the new secret; determining that the  
12 system is uncorrupted if the comparison indicates a match  
13 between the evolved secret and the new secret; and,  
14 determining that the system is corrupted if the comparison  
15 indicate a mismatch between the evolved secret and the new  
16 secret. The term 'in the system' as used herein, includes  
17 elements peripheral to the system but coupled thereto  
18 directly or indirectly.

19 The cryptographic function advantageously comprises a  
20 one-way hash function. The hash function may comprise an  
21 exponentiation function. In an advantageous embodiment of  
22 the present invention, the cryptographic function comprises  
23 a public/private key pair. The initial secret may be  
24 received from a system administrator.

25 Viewing the present invention from another aspect, there is  
26 now provided a data processing system comprising: a  
27 processor; a memory connected to the processor; and,  
28 detection logic connected to the processor and the memory,  
29 the detection logic, in use: providing an initial secret;  
30 binding the initial secret to data indicative of an initial  
31 state of the system via a cryptographic function; recording  
32 state changing administrative actions performed on the  
33 system in a log; prior to performing each state changing  
34 administrative action, generating a new secret by performing

1 the cryptographic function on a combination of data  
2 indicative of the administrative action and the previous  
3 secret, and erasing the previous secret; evolving the  
4 initial secret based on the log to produce an evolved  
5 secret; comparing the evolved secret with the new secret;  
6 determining that the system is uncorrupted if the comparison  
7 indicates a match between the evolved secret and the new  
8 secret; and, determining that the system is corrupted if the  
9 comparison indicates a mismatch between the evolved secret  
10 and the new secret.

11 The present invention also extends to a computer program  
12 element comprising computer program code means which, when  
13 loaded in a processor of a computer system, configures the  
14 processor to perform the aforementioned attack detection  
15 method.

16 The present invention advantageously provides a method for  
17 cryptographic entangling of state and administration in a  
18 data processing system that permits detection of Trojan  
19 horses within the system.

20 In an advantageous embodiment of the present invention, an  
21 initial secret is evolved in a data processing system in an  
22 irreversible manner. The evolution progresses based on  
23 administrative actions in the system. As the evolution  
24 progresses, previous secrets are overwritten. Proof of  
25 knowledge of the evolved secret thus equates to a  
26 cryptographic verification of the history of administrative  
27 action in the system. Thus, if the initial state of the  
28 system is known, the current state can be determined. If the  
29 initial state is known to have been an uncorrupted state,  
30 then an absence of subsequent corruption can be proved. It  
31 can then be determined that the system remains uncorrupted.



1 In a particularly advantageous embodiment of the present  
2 invention, the aforementioned cryptographic entangling has  
3 initialization, update, and proof stages. In the  
4 initialization stage, the system generates an initial secret  
5 and releases binding data that binds to the secret. The  
6 binding data may take different forms depending on  
7 application.

8 In the update stage, the system updates the secret each time  
9 there is an administrative action that could lead to system  
10 corruption such as infiltration of system level Trojan  
11 horses. Examples of such corruption actions include:  
12 updating of system executable code; updating of system  
13 libraries; installation of kernel modules; reading of files  
14 such as those used to store system states during rebooting  
15 operations; alteration of configuration files; alteration of  
16 system run-level codes; and, writing to or reading from  
17 peripheral devices. Each update proceeds in the following  
18 manner. First, a new secret is computed by applying a one  
19 way function to the combination of the previous secret, a  
20 description of the action, and associated context  
21 information. Then, the previous secret and any information  
22 from which it might be derived is completely overwritten or  
23 otherwise erased. A description of the action is then  
24 logged. Finally the action is performed.

25 In the proof stage, the system offers a proof that its  
26 current secret corresponds to the initial secret as it has  
27 evolved according to the record of logged actions.

28 Referring first to Figure 1, a data processing system  
29 comprises a CPU 10, an I/O subsystem 20, and a memory  
30 subsystem 40, all interconnected by a bus subsystem 30. The  
31 memory subsystem 40 may comprise random access memory (RAM),  
32 read only memory (ROM), and one or more data storage devices  
33 such as hard disk drives, optical disk drives, and the like.

1 The I/O subsystem 20 may comprises: a display; a printer; a  
2 keyboard; a pointing device such as a mouse, tracker ball,  
3 or the like; and one or more network connections permitting  
4 communications between the data processing system and one or  
5 more similar systems and/or peripheral devices via a data  
6 communications network. The combination of such systems and  
7 devices interconnected by such a network may itself form a  
8 distributed data processing system. Such distributed systems  
9 may be themselves interconnected by additional data  
10 communications networks.

11 In the memory subsystem 40 is stored data 60 and computer  
12 program code 50 executable by the CPU 10. The program code  
13 50 includes operating system software 90 and application  
14 software 80. The operating system software 90, when executed  
15 by the CPU 10, provides a platform on which the application  
16 software 80 can be executed. The operating system software  
17 90 has a core or kernel 100 of code in which the basic  
18 functions of the operating system software 90 are defined.

19 In an advantageous embodiment of the present invention, the  
20 program code 50 stored in the program code 50 stored in the  
21 memory subsystem 40 also includes detector logic comprising  
22 a cryptographic entangling facility (CEF) 70. The CEF 70 is  
23 associated with a log file 110 also stored in the memory  
24 subsystem 40. The CEF 70 may be separate from the operating  
25 system software 90 as shown or embedded in the operating  
26 system software 90. In a particularly advantageous  
27 embodiment of the present invention, the CEF 70 is embedded  
28 in the kernel 100 of the operating system software 90.

29 In operation, the CEF 70 entangles the state and  
30 administration of the data processing system in a  
31 cryptographic manner. The entangling permits detection of  
32 Trojan horses and the like within the system.

1 As indicated earlier, the administration of the data  
2 processing system includes administrative actions such as  
3 the addition, deletion, or other reconfiguration of the  
4 program code 50 and data 60 recorded in the memory subsystem  
5 40. Each administrative action effectively changes the data  
6 processing system from one state to another.

7 The cryptographic entangling is developed based on an  
8 initial secret. The secret is evolved by the CEF 70 in an  
9 irreversible manner. The evolution of the secret progresses  
10 in steps. Each step corresponds to and is triggered by a  
11 change in state of the data processing system. Whenever an  
12 administrative action is to be performed in the data  
13 processing system, producing a change of state in the data  
14 processing system, there is a corresponding and preceding  
15 step in the evolution of the secret. At each step, the CEF  
16 70 overwrites or otherwise deletes the previous secret from  
17 the memory subsystem 40. The CEF 70 produces a new secret  
18 based on the previous secret together with data indicative  
19 of the corresponding administrative action. The new secret  
20 is recorded by the CEF 70 in place of the previous secret.  
21 The administrative action is then performed. Proof of  
22 knowledge of the evolved secret thus equates to a  
23 cryptographic verification of the history of administrative  
24 action in the data processing system. Thus, if the initial  
25 state of the system is known, the current state can be  
26 determined. If the initial state is known to have been an  
27 uncorrupted state, then an absence of subsequent corruption  
28 can be proved. It can then be determined that the system  
29 remains uncorrupted.

30 Referring now to Figure 2, in advantageous embodiments of  
31 the present invention, the cryptographic entangling has an  
32 initialization stage 200, an update stage 210, and a proof  
33 stage 220.

1 Referring to Figure 3, in the initialization stage 200, the  
2 CEF 70, at step 300 generates an initial secret. The initial  
3 secret is supplied a system administrator via a secure  
4 communication channel. Alternatively, in other embodiments  
5 of the present invention, the initial secret may be entered  
6 to the CEF 70 by the system administrator. At step 310, the  
7 CEF 70 releases binding data. At step 320, the CEF 70 binds  
8 the binding data to the secret. The binding data may take  
9 different forms depending on the data processing system, the  
10 or each application of the data processing system, and trust  
11 mechanisms associated with communication of the secret.

12 Referring to Figure 4, in the update stage 210, the CEF 70  
13 updates the secret each time there is an administrative  
14 action. Specifically, the update is triggered by and  
15 performed is advance of administrative action. As indicated  
16 earlier, examples of such actions include: updating of  
17 system executable code; updating of system libraries;  
18 installation of kernel modules; reading of files such as  
19 those used to store system states during rebooting  
20 operations; alteration of configuration files; alteration of  
21 system run-level codes; and, writing to or reading from  
22 peripheral devices. Each update proceeds in the following  
23 manner. First, at block 400, the CEF 70 computes a new  
24 secret. The new secret is computed by the CEF 70 applying a  
25 one way function to the combination of the previous secret  
26 and data indicative of the administrative action. At block  
27 410, the CEF 70 erases the previous secret, together with  
28 any information from which it might be derived. At block  
29 420, the CEF 70 records data indicative of the  
30 administrative action in the log file 110. At block 430, the  
31 CEF 70 permits execution of the administrative action.

32 With reference to Figure 5, in the proof stage 220, the CEF  
33 70 offers a proof that its current secret corresponds to the  
34 initial secret as it has evolved according to the record of

1 actions contained in the log file 110. Specifically, at  
2 block 500, the CEF 70 retrieves the initial secret. The  
3 initial secret may be retrieved, for example, via a request  
4 for entry of the initial secret by a system administrator.  
5 At block 510, the CEF 70 retrieves the record of  
6 administrative actions from the log file 110 stored in the  
7 memory subsystem 40. At block 520, the CEF 70 evolves the  
8 initial secret based on the record of administrative actions  
9 retrieved from the log file 110. At block 530, the CEF 70  
10 compares the secret evolved in block 520 with its current  
11 secret. If the secrets match, then, at block 550, the CEF 70  
12 reports that the data processing system is still in an  
13 uncorrupted state. If however the secrets do not match,  
14 then, at block 540, the CEF 70 reports that the data  
15 processing system is in a potentially corrupted or otherwise  
16 compromised state.

17 The operation of the CEF 70 in a particularly advantageous  
18 embodiment of the present invention will now be described.  
19 In the following explanation, an ordered collection of  
20 administrative actions in the data processing system is  
21 denoted by  $\{M_i\}$ , where  $i$  is an index. The index may for  
22 example be time. However, such it will appreciated that  
23 intervals between successive administrative actions may be  
24 irregular. The secret known by the CEF 70 at  $i$  is  $S_i$ .  $P(S_i)$   
25 is data that provably binds the data processing system to  
26 the secret  $S_i$ . During the initialization stage 200, the CEF  
27 70 chooses secret  $S_0$  and releases data  $P(S_0)$ . As indicated  
28 earlier, the form of release depends on application. In the  
29 update stage 210, the secret  $S_i$  is updated according to the  
30 mapping  $(S_n, M_n) \Rightarrow S_{n+1}$ . The mapping  $(S_n, M_n) \Rightarrow S_{n+1}$  is a one way  
31 collision resistant cryptographic hash function such as MD-5  
32 or SHA-1. Collision resistance in this context implies that  
33 the pair  $S_n$  and  $S_{n+1}$  together provide a binding to

1 administrative action  $M_n$ . The evolution can thus be  
2 represented by:

$$\begin{array}{c} 3 \quad S_0 \implies^{M_0} S_1 \implies^{M_1} S_2 \dots \implies^{M_{n-1}} S_n \\ 4 \quad \downarrow \dots \downarrow \dots \downarrow \dots \downarrow \\ 5 \quad P(S_0) \rightarrow^{M_0} P(S_1) \rightarrow^{M_1} P(S_2) \dots \rightarrow^{M_{n-1}} P(S_n) \end{array}$$

6 Where the mapping  $P(S_n) \rightarrow P(S_{n+1})$  is an indication that an  
7 administrator can verify that data  $P(S_{n+1})$  is derived from  
8 data  $P(S_n)$  according to administrative action  $M_n$ .

9 In an advantageous embodiment of the present invention, the  
10 initial secret is shared between the data processing system  
11 and the administrators of the data processing system, as  
12 indicated earlier. The computational overhead associated  
13 with this arrangement is advantageously low. However, this  
14 arrangement also assumes that all the administrators are  
15 benign. At the initialization stage 200, the initial secret  
16  $S_0$  is generated by the CEF 70 and provided via a secure  
17 communications channel to the administrators. In the update  
18 stage 210, the CEF 70 employs a collision resistant  
19 cryptographic hash function  $h$  to update the secret, as  
20 follows:

$$21 \quad S_{n+1} = h(\text{"Update"}, S_n, M_n)$$

22 "Update" is a text word included in each update to avoid  
23 spoofing attacks.

24 In the proof stage 220, an administrator supplies the index  
25  $t$  to the CEF 70. In response, the CEF 70 returns  $h(S_n, t)$ . The  
26 initial secret  $S_0$  is known to administrator. In addition,  
27 the history of administrative actions  $\{M_n\}$  is recorded in  
28 the log file 110.  $S_n$  can thus be computed. Because  $h$  is

1 collision free, it is possible to conclude that the data  
2 processing system knows  $S_n$ .

3 In another advantageous embodiment of the present invention,  
4 public key encryption is employed. This has an advantage in  
5 that the administrators of the data processing system need  
6 not trust one another. In the initialization stage 200, the  
7 CEF 70 generates a public/private key pair and publishes the  
8 public key.

9 In the update stage 210, CEF 70 generates, for each  
10 administrative action, a new public/private key pair. The  
11 CEF 70 then signs the combination of the new key and data  
12 indicative of the administrative action  $M_n$  with the  
13 previous key. The CEF 70 records both the signature and the  
14 signed data in the log file. The CEF 70 then erases the old  
15 private key. Only then does the CEF 70 permit the data  
16 processing system to perform the administrative action.

17 In the proof stage 220, the administrator supplies index  $t$ .  
18 In response, the CEF 70 then signs the combination of  $t$  with  
19 the current public key. The history of administrative  
20 actions  $\{M_n\}$  is known. Therefore, the chain of signatures  
21 can be verified. Therefore, the current public key can be  
22 computed. The ability of the CEF 70 to sign a user  
23 influenced message demonstrates that the CEF 70 knows the  
24 private key. Thus, the state of the data processing system  
25 can be confirmed.

26 Another advantageous embodiment of the present invention  
27 involves large prime numbers. In the initialization stage  
28 200, the CEF 70 chooses two large prime numbers  $p_1, p_2$ . The  
29 CEF 70 also chooses an element  $g$  of high order in the group  
30  $G = (\mathbb{Z}/p_1 p_2 \mathbb{Z})^*$ . The CEF 70 picks  $x$  where random  $x \in [1, p_1 p_2]$ . In  
31 addition, the CEF 70 sets  $S_0 = g^x$ . The CEF 70 further picks  $y$

1 where random  $y \in [1, p_1 p_2]$ . The CEF 70 publishes  $g^y$ ,  $g$ ,  $y$ , and  
2 the product  $p_1 p_2$ . The CEF 70 thus binds the data processing  
3 system to  $g^x$ .

4 In the update stage 210, the CEF 70 generates new secrets  
5 based on  $S_{n+1} = (S_n)^{h(n, M_n)} \cdot g$ . This operation is effectively one  
6 way as it is difficult to take roots in the group  $G$ .  
7 Multiplication by  $g$  is added to avoid degenerately small  
8 subgroups.

9 In the proof stage 220, the administrator supplies  $t$ . In  
10 response, the CEF 70 returns  $(S_n)^2 h(t)$ . The history of  
11 administrative actions  $\{M_n\}$  is known from the log file 110.  
12 Modular exponentiation inside the group  $G$  can be performed.  
13 The CEF 70 can compute  $(S_n)^y)^2 h(t)$  by mimicking the updates and  
14 computations that have been performed using  $g^y$  rather than  
15 the system value of  $g^x$ . The value returned raised to the  
16 power  $y$  can be verified.

17 Yet another particularly advantageous embodiment of the  
18 present invention involves Sophie-Germain prime numbers. In  
19 the initialization stage 200, the CEF 70 chooses two  
20 Sophie-Germain prime numbers  $p_1, p_2$ . The CEF 70 also chooses  
21 an element  $g$  in the group  $G = (\mathbb{Z}/p_1 p_2 \mathbb{Z})^x$  with a large prime  
22 order greater than the maximum of the hash function  $h(\cdot)$ . In  
23 addition, the CEF 70 chooses random  $x \in [1, p_1 p_2]$ . The CEF 70  
24 also chooses random  $y \in [1, p_1 p_2]$ . The CEF 70 sets  $S_0 = g^x$ . In  
25 addition, the CEF 70 publishes  $g^y$ ,  $g$ ,  $y$ , and the product  $p_1 p_2$   
26 . In the update stage 210, the CEF 70 generates new secrets  
27 based on  $S_{n+1} = (S_n)^{h(n, M_n)}$ . As indicated earlier, it is difficult  
28 to take roots in the group  $G$ . Thus,  $S_{n+1} = (S_n)^{h(n, M_n)}$  is  
29 effectively one way. Because  $h(n, M_n) < \text{order}(g)$ , no entropy is  
30 lost. Degenerate subgroups are thus avoided. The proof stage



1 220 here is similar to that described in the preceding  
2 example.

3 With reference now to Figure 6, a data processing system as  
4 herein before described with reference to Figure 1 can be  
5 represented as a stack of data processing layers 600-630.  
6 Each layer in the stack forms a foundation of processing for  
7 the next layer in ascending order. At the base level, there  
8 is the hardware layer 630 comprising the CPU 10, and logic  
9 circuitry of the I/O subsystem 20, memory subsystem 40, and  
10 bus subsystem 30. Beyond the hardware layer is the kernel  
11 layer 620, comprising the kernel 100. Beyond the kernel  
12 layer 620 is the operating system layer, comprising the  
13 remainder of the operating system 90. On the operating  
14 system 90 is the application 600, comprising the application  
15 software 80. Other stacked subdivisions of the data  
16 processing system may be apparent to those skilled in the  
17 art. In general, embodiments of the present invention may be  
18 installed in one or more of the data processing layers  
19 600-630. Each installation then protects the layer in which  
20 it is hosted from corruption.

21 In the examples of the present invention herein before  
22 described, all administrative actions producing a change in  
23 state of the data processing system trigger an update of the  
24 secret. However, in other embodiments of the present  
25 invention, only a subset of such actions may trigger an  
26 update of the secret. For example, only actions potentially  
27 leading to infiltration of system level Trojan horses may  
28 trigger an update of the secret.

29 Also, in the advantageous embodiments of the present  
30 invention herein before described, the CEF 70 is implemented  
31 by computer program code executing on the CPU 10 of the data  
32 processing system. It will be appreciated that, in other  
33 embodiments of the present invention, the CEF 70 may be

1 implemented at least partially by hardwired logic circuitry.  
2 In particularly advantageous embodiments of the present  
3 invention, the CEF 70 may be embedded in a trusted subsystem  
4 of the data processing system.

5 Variations described for the present invention can be  
6 realized in any combination desirable for each particular  
7 application. Thus particular limitations, and/or embodiment  
8 enhancements described herein, which may have particular  
9 advantages to the particular application need not be used  
10 for all applications. Also, not all limitations need be  
11 implemented in methods, systems and/or apparatus including  
12 one or more concepts of the present invention.

13 The present invention can be realized in hardware, software,  
14 or a combination of hardware and software. A visualization  
15 tool according to the present invention can be realized in a  
16 centralized fashion in one computer system, or in a  
17 distributed fashion where different elements are spread  
18 across several interconnected computer systems. Any kind of  
19 computer system - or other apparatus adapted for carrying  
20 out the methods and/or functions described herein - is  
21 suitable. A typical combination of hardware and software  
22 could be a general purpose computer system with a computer  
23 program that, when being loaded and executed, controls the  
24 computer system such that it carries out the methods  
25 described herein. The present invention can also be  
26 embedded in a computer program product, which comprises all  
27 the features enabling the implementation of the methods  
28 described herein, and which - when loaded in a computer  
29 system - is able to carry out these methods.

30 Computer program means or computer program in the present  
31 context include any expression, in any language, code or  
32 notation, of a set of instructions intended to cause a  
33 system having an information processing capability to

1 perform a particular function either directly or after  
2 conversion to another language, code or notation, and/or  
3 reproduction in a different material form.

4 Thus the invention includes an article of manufacture which  
5 comprises a computer usable medium having computer readable  
6 program code means embodied therein for causing a function  
7 described above. The computer readable program code means  
8 in the article of manufacture comprises computer readable  
9 program code means for causing a computer to effect the  
10 steps of a method of this invention. Similarly, the present  
11 invention may be implemented as a computer program product  
12 comprising a computer usable medium having computer readable  
13 program code means embodied therein for causing a function  
14 described above. The computer readable program code means  
15 in the computer program product comprising computer readable  
16 program code means for causing a computer to effect one or  
17 more functions of this invention. Furthermore, the present  
18 invention may be implemented as a program storage device  
19 readable by machine, tangibly embodying a program of  
20 instructions executable by the machine to perform method  
21 steps for causing one or more functions of this invention.

22 It is noted that the foregoing has outlined some of the more  
23 pertinent objects and embodiments of the present invention.  
24 This invention may be used for many applications. Thus,  
25 although the description is made for particular arrangements  
26 and methods, the intent and concept of the invention is  
27 suitable and applicable to other arrangements and  
28 applications. It will be clear to those skilled in the art  
29 that modifications to the disclosed embodiments can be  
30 effected without departing from the spirit and scope of the  
31 invention. The described embodiments ought to be construed  
32 to be merely illustrative of some of the more prominent  
33 features and applications of the invention. Other  
34 beneficial results can be realized by applying the disclosed

- 1 invention in a different manner or modifying the invention
- 2 in ways known to those familiar with the art.